upvar manual page - Built-In Commands

tcl.tk/man/tcl/TclCmd/upvar.htm

NAME

upvar — Create link to variable in a different stack frame

SYNOPSIS

upvar ?*level*? *otherVar myVar* ?*otherVar myVar* ...?

DESCRIPTION

This command arranges for one or more local variables in the current procedure to refer to variables in an enclosing procedure call or to global variables. *Level* may have any of the forms permitted for the **uplevel** command, and may be omitted (it defaults to **1**). For each *otherVar* argument, **upvar** makes the variable by that name in the procedure frame given by *level* (or at global level, if *level* is **#0**) accessible in the current procedure by the name given in the corresponding *myVar* argument. The variable named by *otherVar* need not exist at the time of the call; it will be created the first time *myVar* is referenced, just like an ordinary variable. There must not exist a variable by the name *myVar* at the time **upvar** is invoked. *MyVar* is always treated as the name of a variable, not an array element. An error is returned if the name looks like an array element, such as **a(b)**. *OtherVar* may refer to a scalar variable, an array, or an array element. **Upvar** returns an empty string.
The **upvar** command simplifies the implementation of call-by-name procedure calling and also makes it easier to build new control constructs as Tcl procedures. For example, consider the following procedure:

```
proc add2 name {
    upvar $name x
    set x [expr {$x + 2}]
}
```

If *add2* is invoked with an argument giving the name of a variable, it adds two to the value of that variable. Although *add2* could have been implemented using **uplevel** instead of **upvar**, **upvar** makes it simpler for *add2* to access the variable in the caller's procedure frame.

namespace eval is another way (besides procedure calls) that the Tcl naming context can change. It adds a call frame to the stack to represent the namespace context. This means each **namespace eval** command counts as another call level for **uplevel** and **upvar** commands. For example, **info level 1** will return a list describing a command that is either the outermost procedure call or the outermost **namespace eval** command. Also, **uplevel #0** evaluates a script at top-level in the outermost namespace (the global namespace).

If an upvar variable is unset (e.g. **x** in **add2** above), the **unset** operation affects the variable it is linked to, not the upvar variable. There is no way to unset an upvar variable except by exiting the procedure in which it is defined. However, it is possible to retarget an upvar variable by executing another **upvar** command.

TRACES AND UPVAR

Upvar interacts with traces in a straightforward but possibly unexpected manner. If a variable trace is defined on *otherVar*, that trace will be triggered by actions involving *myVar*. However, the trace procedure will be passed the name of *myVar*, rather than the name of *otherVar*. Thus, the output of the following code will be "*localVar*" rather than "*originalVar*":

```
proc traceproc { name index op } {
    puts $name
}
proc setByUpvar { name value } {
    upvar $name localVar
    set localVar $value
}
set originalVar 1
trace variable originalVar w traceproc
setByUpvar originalVar 2
```

If *otherVar* refers to an element of an array, then variable traces set for the entire array will not be invoked when *myVar* is accessed (but traces on the particular element will still be invoked). In particular, if the array is **env**, then changes made to *myVar* will not be passed to subprocesses correctly.

EXAMPLE

A **decr** command that works like **incr** except it subtracts the value from the variable instead of adding it:

```
proc decr {varName {decrement 1}} {
    upvar 1 $varName var
    incr var [expr {-$decrement}]
}
```